

# XP-series DSP Accelerator Cards

XP-15, XP-30, XP-35, XP-100

**User's Guide**  
(r2)

**Version 5.0.0**



Any trademarks or registered trademarks used in this document belong to the companies that own them.

Copyright © 2008, Texas Memory Systems, Inc. All rights are reserved. No part of this work may be reproduced or used in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without permission of the copyright owner.

# Table of Contents

<b>CHAPTER 1 - INTRODUCTION .....</b>	<b>2</b>
1.1 RELATED TEXTS .....	2
1.2 INTERNET SOFTWARE UPDATES .....	2
1.3 REVISION HISTORY .....	2
<b>CHAPTER 2 - OVERVIEW.....</b>	<b>3</b>
2.1 DEVICE ARCHITECTURE .....	3
2.1.1 <i>The XP-100</i> .....	3
2.1.2 <i>The XP-35</i> .....	4
2.1.3 <i>The XP-30</i> .....	5
2.2 XP-SERIES COMPONENTS .....	6
2.3 PROGRAMMING OPTIONS .....	7
<b>CHAPTER 3 - INSTALLATION.....</b>	<b>10</b>
3.1 INSTALLING THE XP-SERIES HARDWARE.....	10
3.1.1 <i>E.S.D. warning</i> .....	10
3.1.2 <i>Installing the XP-Series Card in a computer</i> .....	10
3.1.3 <i>Linux installation</i> .....	11
3.1.4 <i>Installing the Drivers</i> .....	11
3.1.5 <i>Installing the MathXP Library</i> .....	12
3.1.6 <i>Installing the SMT Server</i> .....	12
3.1.7 <i>Installing VPX</i> .....	12
3.2 RUNNING THE XP-SERIES CONFIDENCE TEST .....	13
3.2.1 <i>Short Confidence Test (takes around 20 minutes)</i> .....	13
3.2.2 <i>Looping Confidence Test (runs forever)</i> .....	14
<b>CHAPTER 4 TROUBLESHOOTING .....</b>	<b>15</b>
4.1 OBTAINING SOFTWARE PACKAGE VERSION INFORMATION.....	15
4.2 OBTAINING HARDWARE INFORMATION .....	15
4.3 ERROR MESSAGES .....	16
4.4 HOW TO CONTACT US .....	16
<b>APPENDIX A - THE XP-35 INTRACONNECT .....</b>	<b>17</b>
A.1 HOW DOES THE XP-35 DIFFER FROM THE XP-30? .....	17
A.2 WHAT IS THE XP-35 INTRACONNECT? .....	17
A.3 HOW DOES THE XP-35 INTRACONNECT WORK? .....	17
A.4 CONTROLLING THE INTRACONNECT WITH VPX .....	18
A.5 CONTROLLING THE INTRA-CONNECT WITH POSIX DRIVERS & MATHXP .....	20
A.6 SUMMARY .....	21
<b>APPENDIX B - XP ACCELERATOR CARD FACEPLATES .....</b>	<b>22</b>
B.1 XP-100.....	22
B.2 XP-30.....	22

# Chapter 1 - Introduction

Texas Memory Systems created the XP-Series DSP Accelerator Cards as workstation installable PCI cards that deliver Supercomputer-class vector processing performance. This document describes the XP-Series DSP Accelerator Card installation and set-up process on the following platforms:

Manufacturer	Operating System
Linux x86	Redhat Enterprise 4 & 5
Linux x86_64	Redhat Enterprise 4 & 5

Chapter 2 provides an overview of the XP-Series functionality and programming options. Chapter 3 describes the steps required to install the XP-Series DSP Accelerator Card hardware and software. Chapter 4 contains basic troubleshooting information.

## 1.1 Related texts

Users may find the following Texas Memory Systems documentation useful for reference purposes:

*TM-44 & TM-100 Scientific Math Library Reference Manual*

*VPX User's Guide*

*QuickXP User's Guide*

*XP ECL Interface User's Guide*

## 1.2 Internet software updates

Find the most current Texas Memory Systems software via FTP at <ftp://ftp.texmemsys.com> or via a browser at <http://ftp.texmemsys.com>. Contact Texas Memory Systems for login information.

## 1.3 Revision history

Version	Comments
1.0.1	Initial release
1.0.2	Added TRU64 installation instructions
2.0.0	Standardized instructions across all XP-Series cards
2.0.1	Added VMS XP-15 installation instructions
4.0.0	Added Microsoft Windows 2000 and Windows XP support
4.0.3	Removed support for Redhat 8 and added Enterprise 4
4.1.0	Removed support for VMS
4.2.0	Added support for the XP-35
4.2.3	Support for 64-bit Linux added
5.0.0	Added support for the XP-100 and changed document layout

# Chapter 2 - Overview

The XP-Series DSP Accelerator Cards are used as high-performance vector processors on a single card made for installing in a general-purpose computer's PCI Local Bus card slot. Texas Memory Systems based the XP-100 on the 106-GFLOP TM-100 DSP chip. The rest of the XP-Series is based on the 8-GFLOPS TM-44 DSP chip.

Features Matrix	XP-100	XP-35	XP-30	XP-15
DSP Chips	1 TM-100	2 TM-44	2 TM-44	1 TM-44
VP Device Nodes	2	2	2	1
Total GFLOPs	106	16	16	8
ECL input channels	0	0	2	0
Global memory	2048MB	512MB	256MB	--
Number of processing banks	8	8	8	4
Memory per bank	64MB	64MB	64MB	64MB

## 2.1 Device Architecture

### 2.1.1 The XP-100

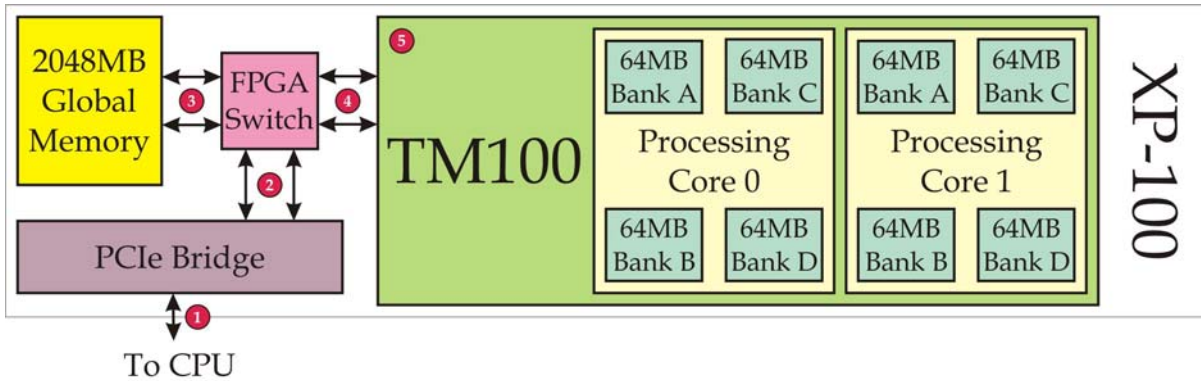


Figure 1 - XP-100 Evaluation Board Architecture

**1 PCIe Bridge / System Memory Bus**

This connection uses a PCI Express x8 bus that is specified to run at 2 GB/sec. Due to 2, the XP-100 will use 1.2 GB/sec of the bandwidth, though performance may vary depending upon your computer's architecture.

**2 Switch / PCIe Bridge Bus**

The FPGA Switch has two independent 700 MB/sec buses to the PCIe Bridge. Both buses run an aggregate bandwidth of 1.2 GB/sec.

**3 Switch / Global Memory Bus**

The FPGA Switch has two independent 2.5 GB/sec buses to Global Memory. Both buses together can run an aggregate bandwidth of 5 GB/sec.

4 **Switch/ TM100 Bus**

The FPGA Switch has two independent 2.5 GB/sec buses to the TM100. Both buses together can run an aggregate bandwidth of 5 GB/sec.

5 **TM100**

Though only a single TM-100 is on the XP-100, the card still has two full processing cores. Each core comes with its own 4 banks of processing memory for a total of 8 banks of processing memory on the board.

### 2.1.2 The XP-35

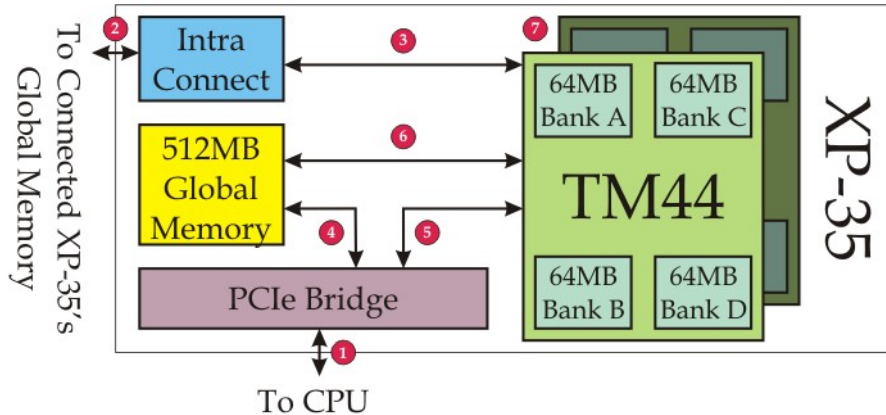


Figure 2- XP-35 Architecture

1 **PCIe XP-35 / System Bus**

This bus uses a PCI Express x8 bus that is specified to run at 2 GB/sec, though performance may vary depending upon your computer's architecture. Due to 4 and 5 this bus will use only 1 GB/sec of that bandwidth.

2 **IntraConnect**

This bus connects to a companion XP-35 and can transport data to and from the local TM-44 and the remote XP-35's Global Memory at 1.2 GB/sec.

3 **Backside IntraConnect**

The IntraConnect is only accessible by the TM-44 node. This allows the TM-44 to use IntraConnected XP-35 Global Memory as its own, effectively doubling the amount of Global Memory from 512 MB to 1 GB.

4 **Global Memory / PCIe Bus**

The backside bus between the Global Memory and the PCIe goes at 1 GB/sec.

5 **TM-44 / PCIe Bus**

The backside bus between each TM-44 node and the PCIe goes at 1 GB/sec.

6 **TM-44 / Global Memory Bus**

Each TM-44 node has a dedicated 1 GB/sec bus to Global Memory. Global Memory can transport 2 GB/sec of data if accessed by both TM-44 nodes simultaneously.

### 7 Dual TM-44 Nodes

The bus design of the XP-35 favors the TM-44 nodes above all else, and it favors both nodes equally. High usage of one TM-44 will not starve the other.

## 2.1.3 The XP-30

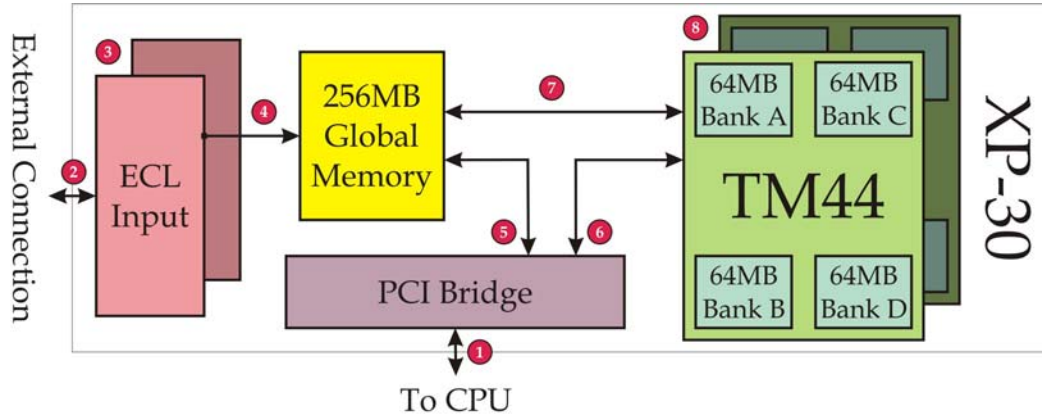


Figure 3 - XP-30 Architecture

### 1 PCI XP-30 / System Bus

The PCI bus is specified to run at just under 500 MB/sec, but most computer architectures will get, at most, 400 MB/sec. Performance will vary depending upon your computer's architecture.

### 2 ECL Input

A single ECL input channel can collect 16bit data as fast as 130MHz. This data bus goes at a total of 260 MB/sec to Global Memory.

### 3 Dual ECL Inputs

Both ECL Inputs can run an aggregate of 520MB/sec to Global Memory.

### 4 ECL In / Global Memory Bus

The backside Global Memory bus can easily sustain the 520 MB/sec aggregate bandwidth of the dual ECL Input channels while keeping both TM-44 nodes fed with data. This allows both TM-44s to process real-time data on the fly.

### 5 Global Memory / PCI Bus

Global Memory has 500 MB/sec of dedicated bandwidth to the PCI bus. Most PCI buses limit this bus, however. The most bandwidth recorded from Global Memory across the PCI bus has been just over 400 MB/sec - which was limited by the System Bus.

### 6 TM-44 / PCI Bus

Each TM-44 has 500 MB/sec of dedicated bandwidth to the PCI bus. Again, like 5, the System Bus limits this throughput.

7 **TM-44/ Global Memory Bus**

Each TM-44 node has a dedicated 1 GB/sec bus to Global Memory. Global Memory can transport 2 GB/sec of data if accessed by both TM-44 nodes simultaneously.

8 **Dual TM-44 Nodes**

The bus design of the XP-30 favors the TM-44 nodes above all else, and it favors both nodes equally. High usage of one TM-44 will not starve the other.

## 2.2 XP-Series components

Global Memory – Provides storage for acquired data and intermediate data

Global Memory is the high-speed memory that connects to the IO, the System Bus, and any TM DSP processing nodes.

Global Memory provides the XP device with large synchronization buffers for incoming data. The XP-30 uses the Global Memory to store acquired data. All XP devices use it to pass data between TM nodes. In an X-Midas application, the VPX primitives use this storage. Global Memory bandwidth always favors the TM nodes and is designed to sustain all on-board nodes with data (i.e. the XP-100 Global Memory bandwidth tops out at 5 GB/sec when accessed by both TM-100 nodes simultaneously).

VP Nodes – Provide the main processing power of the XP-Series

Each of the TM DSP Processing Nodes contains a single DSP core, each with four banks of memory. This DSP core with its four banks of memory is also called a Vector Processing node, or VP node for short. The TM-44 (XP-35, XP-30) has one VP per chip. The TM-100 (XP-100) has two VPs per chip. So, each XP-100, XP-35, and XP-30 has two VP nodes. Each VP acts as an independently controlled device. The user can execute VPX or QuickXP programs written for any XP device on any other XP device.

Texas Memory Systems provides the XP programmer with several hundred permutations of function calls for the TM nodes. The *TM-44 & TM-100 Scientific Math Library Reference Manual* includes detailed specifications for these functions.

ECL Interface (XP-30 only) – Provides 8- or 16-bit dual channel data acquisition

Texas Memory Systems designed the XP-30 ECL with dual, independent ECL channels. Each channel can operate at up to 130MHz on 8 or 16 bit data. The ECL's dual channel architecture diverges from the old architectures where the ECL mimicked dual channel functionality by packing two data channels into a single stream. Instead, the XP-30 has each ECL data stream independently acquired and manipulated. As a result, the XP 30 ECL is more versatile and intuitive.

Users can access each channel directly using the standard `ioctl()` function calls. In addition, VPX includes an X-Midas primitive, `SOURCEUIF`, which can control the XP-30 ECL. For more information on the `SOURCEUIF` primitive, please refer to the *VPX User's Guide*.

Intraconnect (XP-35 only) - Creates a high-speed connection between two XP-35s in the same chassis

Physically, the Intraconnect is a pair of cables that joins two, same-system XP-35s together. It creates a high-speed data conduit that allows all TM44 nodes to share data without going across the PCI bus, or expending host processor resources.

The Intraconnect attaches both XP-35 Mini-SAMs together in one contiguous address space. Each TM44 node on joined cards can access the other card's Mini-SAM just as it accesses its own local Global Memory.

Please read "Appendix A - The XP-35 Intraconnect" on page 17 for more information on the XP-35 Intraconnect.

## 2.3 Programming options

The XP Series DSP Accelerators provide several programming options:

### VPX

Texas Memory Systems designed VPX, which is an X-Midas option tree, to abstract the movement of data in an X-Midas application for the XP-Series DSP Accelerator. VPX handles synchronization issues and manages Global Memory usage. In addition, VPX includes several X-Midas primitives that provide a commonly used set of functionality, such as UBIQ and MFFT.

The MathXP API is a C library that contains mathematical algorithms for VP nodes. These routines include algorithms such as FFT, polyphase filter, and arctangent. Since many X-Midas primitives are written from FORTRAN, VPX provides FORTRAN-callable wrappers to these functions.

Possibly the biggest advantage to using VPX is its ability to abstract synchronization issues inherent to the interaction of multiple VP nodes. Although each node does execute operations in the order that they were issued, synchronization issues exist because VP nodes operate asynchronously from the local processor and other VP nodes. This asynchronous design increases performance by hiding the overhead of calling VP routines on the local processor and by allowing multiple VP nodes to be assigned work and process data simultaneously. VPX simplifies the complexities related to issuing commands to a VP node without waiting for those commands to complete.

VPX abstracts all necessary synchronization details from the programmer. For example, when a primitive calls VPX versions of **GRABX** and **FILAD**, these functions drop synchronization points in the structures and pipe list that VPX maintains. Each time a primitive calls these functions, VPX processes the pipe list to check where the VP is in its execution of commands. When VPX discovers that the data movements are finished, it can safely update the pipe's outbyte or inbyte. Thus, the X-Midas pipe header has an accurate indication of the data read to and written from the pipe, and VPX blocks any primitive trying to read from or write to that pipe until data or space is actually ready.

Additionally, VPX provides a solution for VP-to-VP communication issues. To solve this problem, primitives call routines that act as mailboxes for VP nodes. The following example demonstrates how the message passing typically works: a VP waits for a message to be received; it reads and writes the protected data; it passes a message to another VP; the next VP waits for that message; and so on. The VP nodes themselves handle these messages. The primitive must tell a VP that it is using data from another VP node, and VPX provides the API for accomplishing effective VP-to-VP communication.

More information on VPX can be found in the *VPX User's Guide*.

### QuickXP

QuickXP helps programmers easily harness and manage TMS DSP power in a Linux environment. QuickXP removes DMA bookkeeping while minimizing latency and maximizing processing bandwidth. It runs massively paralleled applications, but codes like a simple program. It facilitates easy prototyping with an intuitive scripting layer, mirroring the programming layer so well that scripted applications can be converted to compiled applications in seconds. Finally, QuickXP is not difficult to learn; a new user can do meaningful work quickly, because all the complication has been hidden and abstracted away.

QuickXP provides a simple C and script interface for the programmer and assists in transferring data into and out of data control abstractions called pipes. Pipes manage data bookkeeping, simplifying the DSP accelerator usage.

As with VPX described in the section above, QuickXP helps abstract synchronization issues inherent to the interaction of multiple VP nodes.

More information on QuickXP can be found in the *QuickXP User's Guide*.

### MathXP

As opposed to the higher-level programming environments of VPX and QuickXP, a programmer can control XP hardware using direct MathXP library calls. This method requires that the programmer manage his or her own data bookkeeping.

The VP programming paradigm should be thought of as calling library functions that are then scheduled for execution in the VP node. Although VP processor execution starts immediately as soon as the first library function call is executed on the host CPU, the CPU's part in this process is quickly completed, and subsequent library function calls are queued at the VP. For many applications a single CPU can schedule hundreds of operations for multiple VP nodes and still have CPU cycles to spare for other activities.

Given this degree of asynchronicity, it is not hard to envision banks of VP nodes, each with its own queue of library functions to execute, operating in parallel with each other and communicating via Global Memory. An XP device can be used to process data in

parallel with other XP cards in the host computer and also with the host computer's CPUs.

More information on MathXP can be found in the *TM-44 & TM-100 Scientific Math Library Reference Manual*.

# Chapter 3 - Installation

## 3.1 Installing the XP-Series hardware

### 3.1.1 E.S.D. warning

XP-Series DSP Accelerator Cards use high-speed electronic components and are susceptible to damage from electrostatic discharge. Any individual handling an XP outside of its anti-static packaging materials should maintain a static-dissipating path to ground. Standard E.S.D. protection procedures, consisting of a conductive wrist strap tied to ground through a resistance of approximately 1-Megohm, provide this protection. Most commercially available E.S.D. protection kits include this resistor as part of the wire that runs from the wrist strap to ground. Note: Damage due to E.S.D. can result in intermittent failures or degradation of performance and may be difficult to diagnose.

Before connecting any cables to an XP, note that unconnected cables can build up static charges through contact with other charged surfaces and with ungrounded individuals. Before using unconnected cables, discharge them by running a finger slowly across all connector pins WHILE EMPLOYING CORRECT E.S.D. PROTECTION PROCEDURES (WRIST STRAP + GROUND CONNECTION).

### 3.1.2 Installing the XP-Series Card in a computer

The XP has the following system requirements:

- Operating system – Linux – RedHat Enterprise 4 & 5
- PCI Bus – PCIe x8 for XP-100
- PCI Bus – PCIe x8 for XP-35
- PCI Bus – PCI (64-bit) for XP-30

First, power off the computer system. Select a suitable vacant PCI slot and remove the faceplate. Insert the XP card firmly into the vacant slot. Lock down the card's faceplate using the screw that previously held in the blank faceplate.

For the XP-100, be sure to connect the 6-pin PCI Express supplementary power connector to the card before powering on the system. Alternatively, use an adapter to go from two 4-pin Molex supplementary power connectors to the 6-pin PCI Express connector on the XP-100. The XP-100 will not operate without the supplemental power.

For ECL cable installation for the XP-30, please refer to the *XP ECL Interface User's Guide*.

Texas Memory Systems provides the latest versions of software via our FTP server at <ftp.texmemsys.com> or via a browser at <ftp://ftp.texmemsys.com>. Contact Texas Memory Systems for login information.

This section describes the steps required to install the software. The account that performs the installation must have root privileges.

### 3.1.3 *Linux installation*

Log on to the computer with system privileges – this is normally root. You need to acquire the following packages from the Texas Memory Systems FTP site:

```
# tms-xp
# mathxp
# cmdserver_smt
# vpx
```

These are just the base name of the packages. The packages names are augmented with various version, platform, and architecture specifications. Some packages have a large number of specific requirements. These requirements are stored in the package name. For example a tms-xp package may be named: tms-xp-5.0.0\_1-504.2.6.9\_67.ELsmp.x86\_64.rpm. This package requires the 2.6.9\_67.ELsmp kernel (RedHat Enterprise 4 Update 6) running on an x86\_64 platform.

Some packages have fewer requirements than others. For example cmdserver\_smt-2.6-285.x86\_64.rpm only requires an x86\_64 platform to properly install.

### 3.1.4 *Installing the Drivers*

The first package that needs to be installed on the system is the tms-xp driver package. As root user install the tms-xp package:

```
# rpm -ihv tms-xp-5.0.0-520.2.6.9_67.ELsmp.x86_64.rpm
Preparing... ##### [100%]
 1:tms-xp ##### [100%]
Loading UDM drivers: [ OK ]
Enabling UDM heartbeat: [ OK ]
Starting ecclogd: [ OK ]
Starting templogd: [ OK ]
No firmware update required for /dev/xp/0 (xp100 n0:v39 n1:v39)
```

The above example assumes the hardware firmware is up-to-date. If the firmware is not up-to-date the package will automatically update the firmware. The package output will look similar to the following:

```
**** Firmware not up-to-date for /dev/xp/0 (xp100)
Node 0 currently at firmware version 14
Node 1 currently at firmware version 14
Firmware RPM wants to install:
Node 0 update file: /opt/texmemsys/tms-
xp/bin/flash/files/xp100/xp100_n0_39.rpd
Node 1 update file: /opt/texmemsys/tms-
xp/bin/flash/files/xp100/xp100_n1_39.rpd
Do you want to UPGRADE node 0 and UPGRADE node 1 (y/n) ? y
/dev/xp/0: node 0: UPGRADE with xp100_n0_39.rpd...Done.
/dev/xp/0: node 1: UPGRADE with xp100_n1_39.rpd...Done.
```

Installing the driver will output information into the `/var/log/messages` file. Please check the file for information if errors occur during the install.

### 3.1.5 *Installing the MathXP Library*

The MathXP package needs to be installed on the system. As root user:

```
# rpm -ihv mathxp-3.1.0-4421.x86_64.rpm
Preparing...      ##### [100%]
   1:mathxp      ##### [100%]

Environment has been modified.
   You will need to restart your session.
```

The MathXP package adds a startup script in the `/etc/profile.d` file which gets called whenever a new shell environment is created. A typical startup script is shown below:

```
% cat /etc/profile.d/tms_mathxp.sh
#!/bin/bash
export LIBTM44=/opt/texmemsys/mathxp
export MATHXP=/opt/texmemsys/mathxp
```

When a new shell is started, both the MATHXP and LIBTM44 environment variables are set. LIBTM44 is a legacy environment variable that is deprecated but still maintained. New applications should use the MATHXP environment variable to locate the MathXP library.

### 3.1.6 *Installing the SMT Server*

The SMT Server is required by VPX and is used to allow Global Memory sharing between multiple VP nodes. VPX can be used without the SMT server, but it is highly recommended that the server be installed. To install the SMT server, as root:

```
# rpm -ihv cmdserver_smt-3.1.0-303.x86_64.rpm
Preparing...      ##### [100%]
   1:cmdserver_smt  ##### [100%]
Starting cmdserver_smt: [ OK ]
```

Once the SMT server has been installed, you may have to modify the configuration file located at `/etc/sysconfig/cmdserver_smt`. Set the SMT\_SIZE to the size of your device's Global Memory (2048M for an XP-100, 512M for an XP-35 and 256M for an XP-30). Then, as root user, restart the SMT server:

```
# service tms.cmdserver_smt restart
Shutting down cmdserver_smt: [ OK ]
Starting cmdserver_smt: [ OK ]
```

### 3.1.7 *Installing VPX*

If you use X-Midas, you will want to install the VPX package. This package writes the VPX option tree to `/opt/texmemsys/vpx`. You may copy the VPX option tree to a more desirable location. To install the VPX package:

```
# rpm -ihv vpx-4.1.0-1806.rpm
```

```

Preparing... ##### [100%]
  1:vpvx ##### [100%]
VPX was installed in directory /opt/texmemsys/vpx

The user name for vpx must already exist on the system.
Enter the user name to use for vpx <xmmgr> -> xmmgr

The group name for vpx must already exist on the system.
Enter the group name to use for vpx <midas> -> midas

Would you like to build VPX in place (Y/n)? y
This will take a few minutes...

```

Notice that the last question allows you to build the VPX option tree at the time of installation. The file ownership will be set according to the user name and group specified at the time of package installation. If you answer with “n” to the build question, please consult the *VPX User’s Guide* for further instructions on building the option tree at a later time. Also refer to that guide for instructions on testing the VPX option tree.

## 3.2 Running the XP-Series confidence test

The XP-Series confidence test is installed with the tms-xp driver package. It is recommend that you run the short confidence test after every installation. If you suspect something is wrong, it is recommended you run the long confidence test. During any point in the confidence test you may hit “q” to end the test prematurely. The XP-Series DSP Accelerator Card should run all tests without any errors. If any errors occur, please contact Texas Memory Systems.

### 3.2.1 Short Confidence Test (takes around 20 minutes)

The XP Monitor (xmon) was installed along with the drivers in the tms-xp package. Enter the monitor:

```

% cd /opt/texmemsys/tms-xp/bin
% ./xmon

```

This will output the following as you enter the monitor:

```

[XMON version 1.46 - Texas Memory Systems, Inc.]

DMA buffer A: Address = 0x2a983c9000  Size = 16 Mbytes
DMA buffer B: Address = 0x2a993cb000  Size = 16 Mbytes

Attempting to open /dev/xp/0 device nodes:
  Found VP node 0.
  Found VP node 1.
  Found SAM memory.

Setting active device to /dev/xp/0.

xmon/top[1]>

```

It will default to run on the first XP device (/dev/xp/0), if you wish to run the monitor on another card just call it with the “-d /dev/xp/#” option. To run the short confidence test:

```
xmon/top[1]> t99
xptype: XP100
T99: Running all basic XP tests (seed = 0xdecafbf0) ...

Testing VP node 0:
T0: SPRAG Test
Error count: 0 (0x0)
T2: Simple SPR Pattern Test (seed = 0xdecafbf00)

Testing SPR:
  Address = 0x0, word count = 0x400
  Pattern 0x0: (size-independent)
:
T16: Bandwidth Test (60 secs, seed = 0xdecafc362)
Aligned on 64 128-bit words
1: icnt= 801 pattern=07:256 addr128=0x1272280 wcnt128=0x36f280  Comp MS
Testing multiple device interaction:
T10[t3]: {0} host=>vp1 host=>vp0 sam=>host
tt10:completed      0 errors
T10[t3]: {0} vp1=>sam vp0=>host sam=>host
tt11:completed      0 errors

T99 total results:
  Error count:      0 (0x0)
0
```

This test will take around 20 minutes to run. It should never stall for more than a minute; it should output test results constantly.

To exit the monitor, simply type “q” and hit return.

### 3.2.2 *Looping Confidence Test (runs forever)*

To run the looping confidence test, follow the same instructions as in the short confidence test above, except type “L99” instead of “T99.” This test will run forever or until an error is detected. Type “q” to stop the test.

# Chapter 4 Troubleshooting

## 4.1 Obtaining Software Package Version Information

To obtain details regarding software package version information, perform the following steps for each package of interest:

```
% rpm -q <package name>
```

For example, the following session shows information for all packages:

```
% rpm -q tms-xp
tms-xp-5.0.0-504.2.6.9_67.ELsmp
% rpm -q mathxp
mathxp-3.1.0-4421
% rpm -q cmdserver_smt
cmdserver_smt-3.1.0-303
% rpm -q vpx
vpx-4.1.0-1806
```

When contacting Texas Memory Systems for support, please have the above information available.

## 4.2 Obtaining Hardware Information

To obtain details regarding the hardware version and configuration, perform the following:

```
% cd /opt/texmemsys/tms-xp/bin/flash
% ./xpepcs_version
```

The output will look similar to the following:

```
=====
XP flash version information
=====

/dev/xp/0  Type: xp100  Serial: <serial number>
-----
Node 0
  date: <date loaded>
  file: <filename>
  user: root
  version: <version number>

Node 1
  date: <date loaded>
  file: <filename>
  user: root
  version: <version number>
```

When contacting Texas Memory Systems for support, please have the above information available.

### 4.3 Error Messages

Under Linux, the file `/var/log/messages` contains any error messages the driver produces. Upon failure of an open, ioctl, or close function, `-1` is returned, and `errno` is set appropriately for the calling program to recognize. Often, the file contains a more complete error description useful for debugging.

### 4.4 How to Contact Us

At Texas Memory Systems, we strive to meet our customers' needs by providing quality products and documentation. This guide should answer most initial questions. Please feel free to contact our Customer Support department with any additional questions, concerns, or comments.

Texas Memory Systems, Inc.  
10777 Westheimer Road, Suite 600  
Houston, TX 77042  
[www.texmemsys.com](http://www.texmemsys.com)  
Tel: (713) 266-3200  
Fax: (713) 266-0332

# Appendix A - The XP-35 Intraconnect

## A.1 How does the XP-35 differ from the XP-30?

There are two obvious differences between the XP-30 and the XP-35. The XP-35 runs over a PCI Express bus (PCIe x8 to be exact) instead of a PCI or PCI64 bus. The XP-35 does not support the XP-30 I/O daughter card.

There are three not-so-obvious differences between the XP-35 and the XP-30. The XP-35 has twice as much Global Memory on it as the XP-30 – 512MB on the XP-35 as opposed to 256MB on the XP-30. The XP-35 has a memory access crossbar whereas the XP-30 does not. This allows for more over-all bandwidth between the VP nodes and Global Memory. The XP-30 has 1GB/sec bandwidth between the TM44 nodes and Global Memory that is shared between the two nodes. The XP-35 has 1GB/sec bandwidth for each TM44 node to Global Memory. If both nodes are fully accessing Global Memory, the XP-35 can run at an aggregate 2GB/sec – twice as fast as an XP-30.

The last not-so-obvious difference between an XP-30 and an XP-35 is the Intraconnect.

## A.2 What is the XP-35 Intraconnect?

Physically, the *Intraconnect* is a pair of cables that joins two same-system XP-35s together. It creates a high-speed data conduit that allows all TM44 nodes to share data without going across the PCI bus or expending host processor resources.

The Intraconnect attaches both XP-35 Global Memories together in one contiguous address space. Each TM44 node on joined cards can access the other card's Global Memory just as it accesses its own local Global Memory.

## A.3 How does the XP-35 Intraconnect work?

When a TM44 node processes data, it can now access that data from the local Global Memory or from the remote Global Memory. Please see Figure 4, below, for a diagram of the Intraconnect.

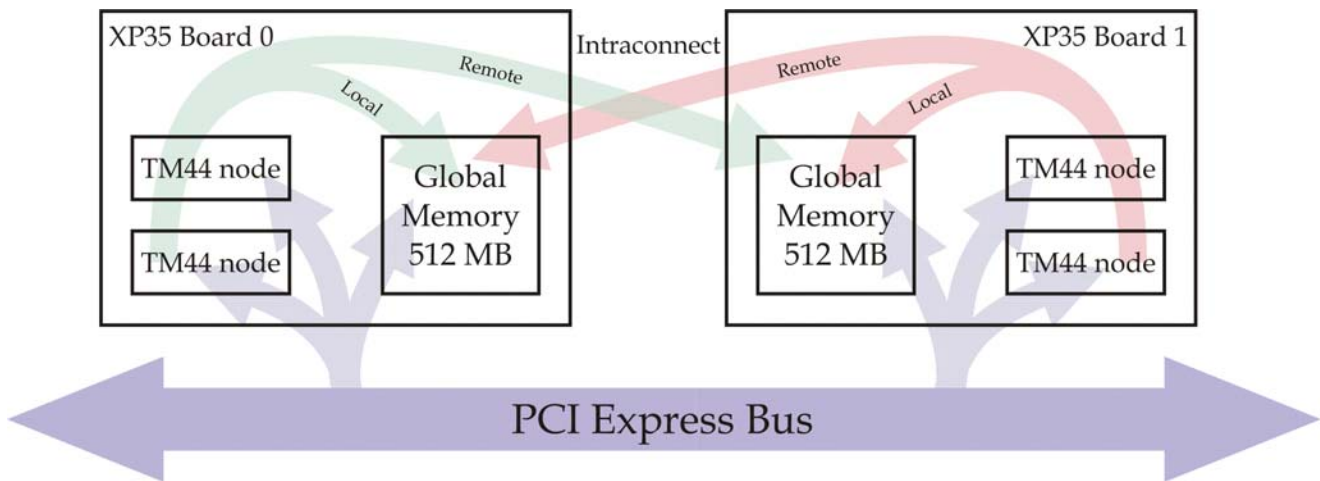


Figure 4 - Diagram of XP-35 Intraconnections

The Intraconnect runs at the same speed of the local connection to Global Memory, so sharing data between two XP-35s comes at no performance cost. A TM44 node can access data on its local Global Memory at 1GB/sec. It can access data across the Intraconnect from a remote Global Memory at exactly the same rate of 1GB/sec.

The XP-35 reserves addresses above 512MB for Intraconnect accesses. The Intraconnect essentially doubles the amount of usable Global Memory to 1024MB and simplifies the way that applications can use it. The XMIDAS option tree, VPX, hides all Intraconnectivity details from the user. **The net result is that operations that were previously illegal are now allowed.**

#### A.4 Controlling the Intraconnect with VPX

The XMIDAS option tree, VPX, controls the remote Global Memory data accesses by simplifying the XP-35 **VPX\_MIRROR** statement. Commands that control dual XP-30 cards in the past are easier to specify since the Intraconnect allows for greater data movement options.

The examples in this section use the following hardware tags:

```

XPMEM0==TMS, MEM, /dev/xp/0/memory,
XPMEM1==TMS, MEM, /dev/xp/1/memory,
XPVP00==TMS, VP, /dev/xp/0/vp0,
XPVP01==TMS, VP, /dev/xp/0/vp1,
XPVP10==TMS, VP, /dev/xp/1/vp0,
XPVP11==TMS, VP, /dev/xp/1/vp1,

```

Currently, the following set of instructions work on both an XP-30 and an XP-35.

```

vpv_mirror _src _src(ps=32m) (sam="XPMEM0")
vpv_mfft/vplist="XPVP00,XPVP01" _src _out(ps=32m) (sam="XPMEM0")
vpv_mirror _out _out(ps=8m)

```

Here VPX mirrors the XMIDAS pipe, `_src`, into the Global Memory on XP0, runs its vector operation, and mirrors the results from XP0 to an output XMIDAS pipe. The **VPX\_MFFT** DSP operation only runs on a single card's TM44 nodes, specified by the `/vplist` switch.

To run the same **VPX\_MFFT** DSP operation on four TM44 nodes using two XP-30 cards, the command becomes a bit more complicated:

```
vpX_mirror _src _src(ps=32m)(sam="XPMEM0, XPMEM1")
vpX_mfft/vplist="XPVP00,XPVP01,XPVP10,XPVP11" _src
    _out(ps=32m)(sam="XPMEM0, XPMEM1")
vpX_mirror/tl=2 _out(sam="XPMEM0, XPMEM1") _out(ps=8m)
```

The bandwidth demands on this code are higher than using a single card. The X-MIDAS pipe is DMAed into both XP-30 Global Memories – doubling the DMA requirements – despite each card only needing a portion of the input data. The output data is mirrored into an X-MIDAS pipe taking half of the results from the first XP-30's Global Memory and the other half from the second's, specified by the "/tl=2" switch.

Using XP-35s joined via an Intraconnect, the following code mirrors the XMIDAS \_src pipe into one card's Global Memory. The **VPX\_MFFT** DSP operation works on the data located on that card across all four TM44 nodes and then the data is mirrored out of the cards from the second XP-35's Global Memory. The resulting operation is a lot cleaner, both in code and in throughput requirements.

```
vpX_mirror _src _src(ps=32m)(sam="XPMEM0")
vpX_mfft/vplist="XPVP00,XPVP01,XPVP10,XPVP11" _src _out(ps=32m)(sam="XPMEM1")
vpX_mirror _out _out(ps=8m)
```

Let's look at the VPX demonstration program, `demo_ap2.txt`, located in `vpX/mcr`. The program runs the **VPX\_UBIQ** primitive across four TM44 nodes on two XP-30s. Here is the heart of the program:

```
res a:wr_sam_devs "XPMEM0, XPMEM1"
res a:rd_sam_devs "XPMEM0, XPMEM0, XPMEM1, XPMEM1"
:
xpipe/setup/controls on
:
vpX_mirror _srcal _srcal(ps={SAM_PipeSize})(sam="{wr_sam_devs}")
vpX_mirror _srcb1 _srcb1(ps={SAM_PipeSize})(sam="{wr_sam_devs}")

vpX_ubiq/fftp=pack_mode/ctag=(5:p_speed)/nvp=4/reps={num_avg} &
    _srcal ,, ^{fftsize} HANNING ^{ol_val} ^{coupling} ^{num_avg} &
    _srcb1 ,, _out(ps={SAM_PipeSize})(sam="{wr_sam_devs}")

vpX_mirror/tl=1 _out(sam="{rd_sam_devs}") _out(ps={host_PipeSize})
:
xpipe off
```

The program is writing the `_srcal` and `_srcb1` pipes to both XP-30 *Global Memories*. Each board gets identical copies of the vectors. The **VPX\_UBIQ** call then generates partial result vectors on each XP-30, since each board processes half the data – it then copies and interleaves that data back to the host.

Here is the same code written for the XP-35:

```

res a:wr_sam_devs "XPMEM0"
:
xpipe/setup/controls on
:
vpx_mirror _srcal _srcal(ps={SAM_PipeSize})(sam="{wr_sam_devs}")
vpx_mirror _srcbl _srcbl(ps={SAM_PipeSize})(sam="{wr_sam_devs}")

vpx_ubiq/fftp=pack_mode/ctag=(5:p_speed)/nvp=4/rep={num_avg} &
_srcal ,, ^{fftsize} HANNING ^{ol_val} ^{coupling} ^{num_avg} &
_srcbl ,,_out(ps={SAM_PipeSize})(sam="{wr_sam_devs}")

vpx_mirror _out _out(ps={host_PipeSize})
:
xpipe off

```

Looking closely at the snippet, you can see that the code has been shortened and simplified. Gone is the double memory specification for `wr_sam_devs`. Gone is the four-memory specification of `rd_sam_devs`. Gone is the `"/t1=1"` switch that made the host interleave results from two separate Global Memories. The resulting code is cleaner; the resulting algorithm is faster and more efficient.

The Intraconnect really shines when data has to be shared across multiple TM44 nodes. The XP-30 allows data to be easily shared between two TM44 nodes, whereas two XP-35s with an Intraconnect can share data between four TM44 nodes just as easily. Code-wise, the operation is the same since VPX hides all data hops through host memory. With an XP-35 Intraconnect, however, the hidden hop does not result in a performance hit since the no hop is actually required.

## A.5 Controlling the Intra-connect with POSIX drivers & MathXP

To control Intraconnect accesses using POSIX driver commands, simply offset the Global Memory buffers by the size of the Global Memory. If the address exists outside of the local Global Memory's address range, the hardware will read the Global Memory attached via the Intraconnect. For example:

```

tms_xp_sam_xfer_t xfer;
unsigned int memsz8;
int memfd, vpdf, sendbuf[BUFSIZE32], recvbuf[BUFSIZE32];

/* Open the global memory on card 0 */
memfd = open("/dev/xp/0/memory", O_RDWR); CK;

/* Open TM44 node on card 1 */
vpdf = vp_open("/dev/xp/1/vp0", O_RDWR); CK;

/* Read the size of Global Memory (always 512MB) */
st = ioctl(memfd, TMS_XP_IOC_SAM_SIZE, &memsz8); CK;

/* Initialize the data to be transmitted to card 0 */
for (ii=0 ; ii<BUFSIZE32 ; ii++) sendbuf[ii] = ii;

```

```

/* Transfer the data into card 0's global memory */
xfer.mem_addr8 = (size_t)sendbuf;
xfer.sam_addr8 = 0;
xfer.cnt8      = BUFSIZE32*4;
xfer.swap     = 0;
xfer.dir      = TMS_XP_XFER_TO_DEV;
st = ioctl(memfd[0], TMS_XP_IOC_SAM_XFER, &xfer); CK;

/* Setup global memory buffer on card 1 - this is the same data written to
   card 0, but since it is offset by the global memory size, memsz8, it
   will actually read from card 0. */
VBUF_SETUP(&msbuf, 1, VMATH_SAM_MEMORY, memsz8, VU32);

/* Setup a TM44 buffer in card 1 */
VBUF_SETUP(&bdbuf, vpdf, VP_RAM_A_MEM, 0, VU32);

/* DMA from card 0's global memory to card 1's TM44 via the intraconnect */
st = vcopy(&msbuf, &bdbuf, 0, BUFSIZE32); CK;

/* Initialize the receive buffer */
memset(mybuf, 0, sizeof(recvbuf));

/* Read the data from card 1's TM44 node */
VBUF_SETUP(&htbuf, 0, VMATH_SP_MEMORY, recvbuf, VU32);
st = vcopy(&bdbuf, &htbuf, 0, BUFSIZE32); CK;

/* Wait for the operation to complete */
st = vp_sync(vpdf, 2000); CK;

```

The above program moves data from a host buffer into one XP-35's Global Memory. It then moves the data across the Intraconnect into the second XP-35's TM44 node. Finally it moves the data from the second XP-35's TM44 node back into the host buffer.

## A.6 Summary

With two XP-35 boards, XMIDAS algorithms can run 2 to 3 times faster than a single XP-30 with no extra host bandwidth or memory resources required – on algorithms that are I/O bound, the XP-35s can execute at an even higher rate. The PCI Express bus provides four to eight times the bandwidth to an XP-35 over that of the PCIX bus to the XP-30. This combined with the Intraconnect, the memory crossbar, and the twice-as-large Global Memory makes the XP-35 the outstanding performer in an already impressive product lineup.

# Appendix B - XP Accelerator Card Faceplates

## B.1 XP-100

The following illustration shows the XP-100 faceplate:

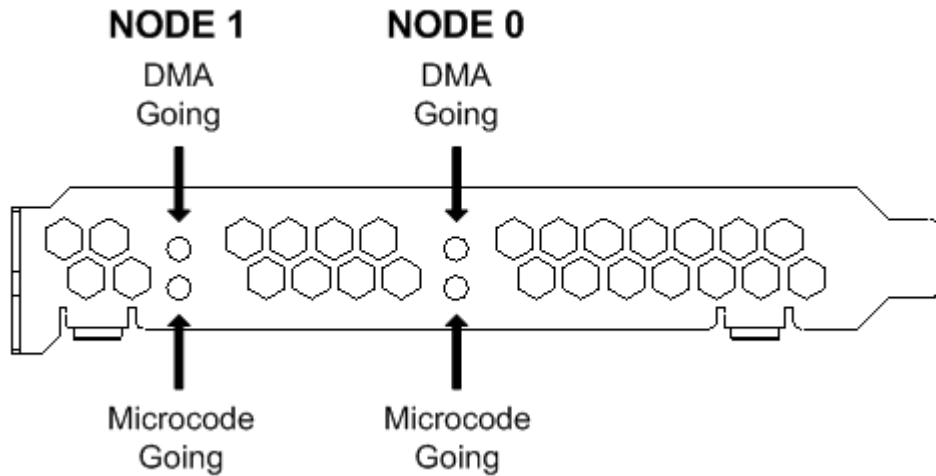


Figure 5 - XP-100 Faceplate

The arrows shown indicate the 4 LEDs on the faceplate, and "Node" above indicates a VP node. The LEDs convey various pieces of status information for the XP-100. The LEDs will be on and green when idle and blink to indicate activity. If the TM100 detects errors, the LED's turn to an orange color.

## B.2 XP-30

The following illustration shows the XP-30 faceplate:

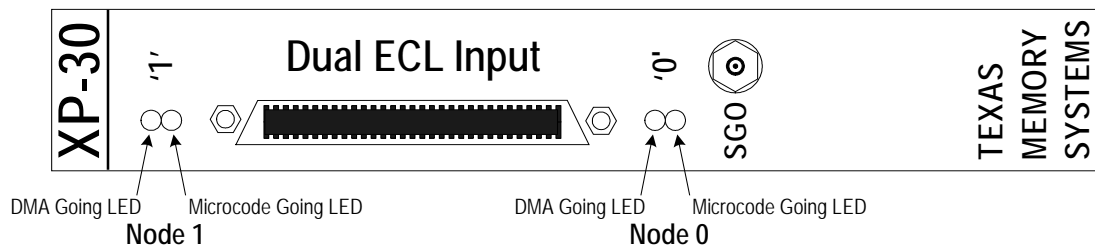


Figure 6 - XP-30 Faceplate

Each XP-ECL-IN provides 2 external connections: (1) one 68-pin connector for data labeled 'Dual ECL Input,' and (2) one SMB connector labeled SGO. The arrows shown indicate the 4 LEDs on the faceplate, and "Node" above indicates a VP node.